WG Working Group
Internet-Draft
Intended status: Informational
Expires: 21 November 2026

B. Curtis
MyTerms

20 May 2026

# MyTerms Contract Negotiation Protocol (MCNP): Human and machine-readable agreements
draft-curtis-myterms-01

## Abstract

This document covers the technical requirements of Verifiable Contractual Agreements (VCAs) between individuals and the entities that engage on a network as defined in IEEE7012. It describes how individuals, acting as first parties, can proffer their privacy requirements as contractual terms and arrive at agreements recorded and kept by both sides. This includes the hosting format for contracts, negotiation of contracts, signing of contracts, and auditing of contracts.

## About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://codeberg.org/myterms/ietf/src/branch/main/draft-curtis-myterms-01.xml. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-curtis-myterms/.

Discussion of this document takes place on the WG Working Group mailing list (mailto:WG@example.com), which is archived at https://example.com/WG.

Source for this draft and an issue tracker can be found at https://codeberg.org/myterms/ietf.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

## Copyright Notice



## Table of Contents

# 1. Introduction

## 1.1. Purposes of MyTerms

The purpose of the [IEEE7012] standard, otherwise known as MyTerms, is to provide individuals with means to proffer their own terms respecting personal privacy in ways that can be read, acknowledged and agreed to by machines operated by others in the networked world. In a more formal sense, the purpose of the standard is to enable individuals to operate as first parties in agreements with others, mostly organizations, operating as second parties.

In this methodology, agreements shall be chosen from a registry of standard-form agreements in a roster kept by an independent and neutral non-business entity. Computing devices and software performing as agents for both first and second parties shall engage using the protocol defined in this document. The first party shall point to a preferred agreement, or a set of agreements, from which the second party shall accept one. Party-to-party negotiations over agreements in any of these contracts or other agreements are outside the scope of this standard. If both parties agree, the chosen contract or agreement shall be signed electronically by both parties' agents, and a matching record shall be kept in a way that can be retrieved, audited, or disputed, by either side if necessary, at some later time.

## 1.2. Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

A "PERSON AGENT" is defined as any system, such as a web browser, mobile application, AI/MCP agent, or any other tooling, that negotiates MyTerms agreements on behalf of a consuming party, otherwise known as the first party in [IEEE7012]. An "ENTITY AGENT" is defined as any system, such as a web server, operating system, AI/

MCP agent, or any other tooling, that negotiates MyTerms agreements on behalf of a providing party, otherwise known as the second party in [IEEE7012].

## 2. Hosting agreements

### 2.1. Agreement types

There are three types of agreements in MyTerms:

* *Relationship agreements:* Agreements that pertain to the overall relationship between the PERSON and the ENTITY, such as for service delivery or data portability.

* *Personal Data Contribution agreements:* Agreements that pertain to one-time data use or exchange.

* *Legal agreements:* Agreements that contain the legal language enforcing the Relationship and Personal Data Contribution agreements.

Depending on the interaction type between the PERSON AGENT and ENTITY AGENT, a specific type, or set of a type, of agreements MAY be signed.

### 2.2. Agreement levels and codes

Every agreement MUST have an associated code that represents the agreement. Each relationship agreement’s code MUST represent the level of restriction on the agreement.

Restriction levels SHOULD move linearly from most restrictive on data use to least restrictive on data use, meaning the fewer characters in the code represent more restrictions on data use. An example of this would be:

| Code | Level | Meaning |
|---|---|---|
| SD-BASE | 1 | Allows service delivery |
| SD-BASE-A | 2 | Allows service delivery and analytics |
| SD-BASE-AT | 3 | Allows service delivery, analytics, and tracking |
| SD-BASE-ATP | 4 | Allows service delivery, analytics, tracking, and profiling |
| SD-BASE-ATP3 | 5 | Allows service delivery, analytics, tracking, profiling, and 3rd-party sharing |

Table 1

This structure simplifies the machine negotiation when multiple agreements are in play.

## 2.3. Ensuring tamper-proof agreements

To ensure hosted agreements are tamper-proof they are built as Verifiable Contractual Agreements (VCAs), meaning that when a party signs an agreement, there is proof that the content of the agreement has not been altered since the signature took place. Agreements MUST be hosted in Markdown (MD) format with a hash of the content in the URL.

Agreements MUST also be hosted in a web-browser friendly format, for instance using Markdown to HTML conversion tools. Web-browser friendly versions MUST contain links to the raw Markdown versions and MUST contain links to the machine-readable versions.

URL formats for web-browser friendly HTML agreements SHOULD be in the format of:

https://<domain>.<tld>/<first letter of type>/<code>

URL formats for Markdown agreements SHOULD be in the format of:

https://<domain>.<tld>/<first letter of type>/<code>/<hash>.md

URL formats for machine-readable agreements SHOULD be in the format of:

https://<domain>.<tld>/<first letter of type>/<code>/<hash>.json

Web-browser friendly HTML agreements MUST be generated directly from their markdown counterparts, and MUST include links to:

* The hashed markdown version
* The HTML legal agreement, if it is not the legal agreement
* The hashed markdown legal agreement, if it is not the legal agreement
* Any vocabulary definitions used in the agreement, such as the [DPV]
* The machine-readable JSON version

## 2.4. Machine readable agreements

Machine readable agreements MUST be JSON. The format of the JSON MUST be:

```
{
  "version": <machine-readable agreement version>,
  "references": [
    "<link to hashed agreements>"
  ],
  "agreementId": "<UUID representing this agreement>",
  "created": <epoch time>,
  "ids": [
    "<optional DID id>"
  ],
  "permitted": [
    "<purpose>"
  ],
  "prohibited": [
    "<prohibition>"
  ],
  "validRoles": [
    "<role>"
  ]
}
```

Machine readable agreements MUST support signing via DIDs as per the [DID] W3C recommendation. DID ids within the ids array of the agreement MAY include the ENTITY AGENT's DID id, and if it is included they MUST sign the agreement before it is considered valid.

Multiple human-readable (hashed markdown) agreements MAY be grouped into the references block, and SHOULD include a vocabulary set such as the [DPV], the primary agreement, and an overlapping legal agreement.

The permitted array MUST contain a representation of any expressly allowed actions within the agreement content. The prohibited array MUST contain a representation of any expressly prohibited actions within the agreement content. The validRoles array MUST contain a list of roles or parties as defined within the agreement content, such as that of entity, user, and/or third-party.

Machine readable agreements MAY be signed by multiple PERSON AGENTs, however agreement UUIDs MUST be unique per unique JSON record.

## 2.5. Retrieving available agreements

When a PERSON AGENT or ENTITY AGENT wishes to retrieve a list of codes and their corresponding agreements from the agreement hosting entity, a publicly-accessible API endpoint SHOULD be provided for accessing this information.

The URL format for the API endpoint to set preferences MAY be:

https://<domain>.<tld>/api/v1/myterms/agreements

Retrieving agreements MUST be completed via a GET, and the response MUST be in the format of:

```
{
  "agreements": [
    {
      "type": "<relationship|personal_data_contribution|legal>",
      "agreements": [
        "title": "<agreement title>",
        "code": "<agreement code>",
        "url": "<url to human readable HTML agreement>",
        "jsonUrl": "<url to machine-readable agreement>",
      ]
    }
  ]
}
```

When an agreement is rotated out for a new version, the hashed markdown and machine-readable JSON versions of that agreement MUST continue to remain available for previous lookups, and the agreement MUST be removed from the endpoint results.

## 2.6. Configuring agreement preferences

A user with a role of PERSON AGENT MUST be able to select a set of individual agreements they would agree to during the negotiation. When a PERSON AGENT wishes to set the code that represents the preferences that have been selected by the user, an API endpoint SHOULD be provided for saving this information.

The URL format for the API endpoint to set preferences MAY be:

https://<domain>.<tld>/api/v1/myterms/prefs

Setting or updating agreement preferences SHOULD be completed via a JSON POST, and the POST body SHOULD be in the format of:

```
{
  "agreements": [
    "<agreement code>"
  ]
}
```

Multiple agreement codes can be included, and the system should associate those selections to the appropriate agreements. Additional information MAY be included in the JSON content.

Authentication MUST be used for this endpoint, and SHOULD follow standard best-practices for web authentication.

## 2.7. Retrieving agreement preferences

When a PERSON AGENT wishes to retrieve the codes that represents the preferences that have been selected from the agreement hosting entity, an API endpoint MUST be provided for accessing this information.

The URL format for the API endpoint to set preferences MAY be:

https://<domain>.<tld>/api/v1/myterms/prefs

Retrieving agreement preferences MUST be completed via a GET. The response MUST be in the format of:

```
{
  "agreements": [
    "<agreement code>"
  ]
}
```

Authentication MUST be used for this endpoint, and SHOULD follow standard best-practices for web authentication.

# 3. Agreement negotiation mechanisms

## 3.1. Client preference delivery

### 3.1.1. On-demand proposal of agreements

In this method, the PERSON AGENT to ENTITY AGENT negotiation occurs via the exchange of data over HTTP/S, where the ENTITY AGENT delivers an endpoint location to the PERSON AGENT, and the user must take an action before terms are delivered. This eliminates any sources of pervasive monitoring as clients can ignore the request from an ENTITY AGENT. This method SHOULD be used for interactions such as signing Terms of Use, or agreeing to actions in a website or application.

First, the ENTITY AGENT hosts a URL containing a JSON of the details of what agreements they support. The JSON format SHOULD be:

```
{
  "endpoint": "https://<domain>.<tld>/api/v1/myterms/put",
  "agreements": [
    {
      "type": "<relationship|personal_data_contribution>",
      "required": <boolean>,
      "url": "<url of machine-readable JSON agreement>"
    }
  ]
}
```

Any required agreements in the array MUST be accepted by the PERSON AGENT before they can proceed. The JSON record MAY contain additional data and information by use by the providing system. The endpoint variable MUST represent the location that signed agreements can be submitted to the ENTITY AGENT from the PERSON AGENT.

Next, the PERSON AGENT MUST validate that the URLs of all agreements from the ENTITY AGENT are from the same hosted subdomain as the subdomain of the configured PERSON AGENT. If they do not match, the negotiation MUST be rejected and the user MUST be informed of the discrepancy.

Lastly, the PERSON AGENT retrieves the machine-readable JSON agreements to begin the negotiation with the ENTITY AGENT.

#### 3.1.2. Continuous proposal of agreement

In this method, the PERSON AGENT to ENTITY AGENT negotiation occurs also over HTTP/S via the exchange of HTTP headers, but this time on every request. This method SHOULD be leveraged for interactions that do not have the capabilities or process abilities to support on-demand agreement delivery, such as for up-front cookie preference negotiation in browsers.

First, when a PERSON AGENT makes a request to a particular web page or service endpoint that is MyTerms enabled, the ENTITY AGENT will return a response HTTP header of a MyTerms request endpoint that MUST reference a JSON file of the same format as On-demand proposal of agreements:

X-MyTerms-Agreements: <entity agreement endpoint url>

When encountering a MyTerms response header request, a PERSON AGENT MUST alert the user to the request for them to allow or deny, unless the user has previously approved that PERSON AGENT to accept all MyTerms requests from that domain. A system MUST NOT allow users to accept all MyTerms requests for all domains. This alert MUST contain full host name contained within the header, including domain, top level domain, and subdomain if one exists.

From here, the PERSON AGENT SHOULD follow the same pattern as if it had retrieved the JSON in the On-demand proposal of agreements flow.

### 3.2. Mitigating Pervasive Monitoring

As per [RFC7258], to mitigate Pervasive Monitoring (PM) and thus decrease the ability of entities to leverage MyTerms user preferences for fingerprinting, PERSON AGENTS MUST NOT deliver information beyond the signed agreements unless tied to another flow outside of this standard that properly notifies that user of a data exchange. An acceptable example would be including an OIDC login with the signing of a Terms of Use, where the OIDC flow provides an indication of what data is being exchanged.

### 3.3. Agreement negotiation algorithm

Once the PERSON AGENT receives the acceptable agreement selections from the ENTITY AGENT, it is the PERSON AGENT's job to handle the negotiation. At this point the PERSON AGENT should compare it's acceptable agreements with those of the ENTITY AGENT, and confirm that the user has allowed the agreements required by the ENTITY AGENT.

If the PERSON AGENT provides terms that are less restrictive than the ENTITY AGENT requires, the PERSON AGENT MAY sign a less restrictive agreement.

The following chart demonstrates how these negotiations MUST occur. In these examples the same codes from above are used.

| PERSON provides | AGENT requires | AGENT supports | Result |
|---|---|---|---|
| SD-BASE | SD-BASE | | Proceed to signing and delivery of SD-BASE |
| SD-BASE | SD-BASE | SD-BASE-A | Proceed to signing and delivery of SD-BASE |
| SD-BASE-A | SD-BASE | SD-BASE-A | Proceed to signing and delivery of SD-BASE |
| SD-BASE,SD-BASE-A | SD-BASE | SD-BASE-A | Proceed to signing and delivery of SD-BASE-A |
| SD-BASE | SD-BASE-A | | Use must be notified they must sign SD-BASE-A to continue |

Table 2

#### 3.3.1. Requiring multiple agreements

If an entity requires multiple agreements to be signed, the PERSON AGENT follows the same pattern as above, and SHOULD alert the user that certain agreements must be signed before proceeding. In any displays offering up the ability to select these agreements, the PERSON AGENT MUST NOT pre-check any options unless the user has already set these options in their default preferences.

## 4. Signing agreements

### 4.1. Levels of attestation

When PERSON AGENTs sign agreements, there are 3 levels of attestation, one of which MUST be attested to for the agreement to be valid. The 3 levels are:

| Level | Title |
|---|---|
| 1 | Server trusted remote attestation |
| 2 | Cryptographic attestation |
| 3 | Auditable cryptographic attestation |

Table 3

As the level increases, the security and verifiability of the attestation increases.

#### 4.1.1. Level 1: Server trusted remote attestation

In this level, ENTITY AGENTs MUST give PERSON AGENTs a checkbox that when checked, represents the signing of an agreement. ENTITY AGENTs MUST default the checkbox to unchecked.

Under this level of attestation, PERSON AGENTs are trusting that ENTITY AGENTs will act honestly and transparently around storing and abiding by their preferences, with no third-party auditing.

#### 4.1.2. Level 2: Cryptographic attestation

In this level, ENTITY AGENTs MUST give PERSON AGENTs a method of cryptographically signing an agreement. Agreements MUST be signed utilizing a private key that represents the PERSON AGENT. This private key SHOULD be fully controlled by the end user, and not available to the ENTITY AGENT in any way. This private key MAY be owned by the ENTITY AGENT or an agent, if that ENTITY AGENT or agent was given a capability delegation from the user, and that delegation MUST be cryptographically signed and provided via a DID.

Signing an agreement MUST occur using an EdDSA JWT and a private key associated with any public key in the verification method of the signing DID. For example, if the signing DID document is represented by the following FedID enabled DID:

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://didspec.myterms.info/v2/ctx.jsonld"
  ],
  "capabilityDelegation": [
    {
      "id": "did:myterms:fedid.myterms.info:GDj...",
      "type": "myterms",
      "archiveServers": [ "https://archive.myterms.info" ]
    }
  ],
  "created": "2025-10-30T16:00:14Z",
  "deactivated": false,
  "id": "did:myterms:fedid.myterms.info:iuK...",
  "recoveryHash": "d0a634b07cea22a9e3865eb5598e0486636bb772976bf93d",
  "service": [
    {
      "id": "did:myterms:fedid.myterms.info:5c7...",
      "serviceEndpoint": "https://fedid.myterms.info",
      "type": "login"
    }
  ],
  "shortName": "person@fedid.myterms.info",
  "updated": "2025-10-30T16:00:14Z",
  "verificationMethod": [
    {
      "controller": "did:myterms:fedid.myterms.info:iuK...",
      "created": "2025-10-30T16:00:14Z",
      "deactivated": null,
      "id": "#55b72a71d2431bd5eb10b347f1b272da",
      "key": "9ud_btCMlYQRHkTyyKMNtC48avmC60fCZhWNIbOsNi0",
      "type": "device"
    }
  ],
  "version": "2.0"
}
```


Then the private key associated with the public key 9ud_btCMlYQRHkTyyKMNtC48avmC60fCZhWNIbOsNi0 should be utilized. As this DID document also contains a capability delegation that MUST be of type myterms, any keys in the verification method array of that DID document may sign on behalf of the user.

The agreement MUST be signed via JWS following [RFC7515] after following [RFC8785] for canonicalization to deterministically sort by recursively ordering the keys alphanumerically and sorting any arrays alphanumerically. The resulting JSON data MUST be submitted via POST

to the entity endpoint for submitting signed agreements. The format for the posted data MUST match the provided example, where the public key provided is the public key paired with the private key that was used to sign from the verification method array.


```
{
  "agreement": {
    "agreement": {
      "version": 1,
      "references": [
        "https://<url>/a/74d...33d"
      ],
      "agreementId": "1df77ef2-e3c6-4f2b-859d-379cb9874d78",
      "created": 1760621627589,
      "ids": [
        "did:myterms:fedid.myterms.info:YTc..."
      ],
      "permitted": [
        "service-analytics",
        "service-delivery"
      ],
      "prohibited": [
        "profiling",
        "third-party-analytics",
        "third-party-sharing",
        "tracking"
      ],
      "validRoles": [
        "entity",
        "third-party",
        "user"
      ]
    },
    "signature": {
      "version": 1,
      "id": "did:myterms:fedid.myterms.info:iuK...",
      "signedOn": 1761841201,
      "type": "JWS/JCS",
      "jws": "eyJ...ICg"
    }
  },
  "publicKey": "9ud_btCMlYQRHkTyyKMNtC48avmC60fCZhWNIbOsNi0"
}
```


Under this level of attestation, PERSON AGENTs are trusting that ENTITY AGENTs will act honestly and transparently around storing and abiding by their preferences, with no third-party auditing, but gain the security of a referenced cryptographic signature.

#### 4.1.3. Level 3: Auditable cryptographic attestation

In this level, the same signing method MUST be used for agreements, however an additional zero-knowledge audit record MUST be provided. This allows a third-party to validate that the agreement was appropriately signed without having any knowledge of what the agreement contains.

After the agreement is signed, an audit record MUST be generated from the top-level agreement object, which consists of the original agreement and it’s signature. This object MUST be deterministically sorted, and a SHA256 digest MUST be created. The format of the audit record MUST include:

* A context of keys used in the audit record
* A reference to the agreement ID that was signed
* A zero-knowledge representation of the signed agreement
* When the audit record was created

Under this level of attestation, PERSON AGENTs are trusting that ENTITY AGENTs will act honestly and transparently around storing and abiding by their preferences, with the security of a referenced cryptographic signature and a third-party that can audit that signature was performed without modification to the agreement.

### 4.2. Retrieving signed agreements

For services that store signed agreements on behalf of PERSON AGENTs and ENTITY AGENTs, an API endpoint SHOULD be provided for retrieving signed agreements. The URL format for the API endpoint to retrieve signed agreements MAY be:

https://<domain>.<tld>/api/v1/myterms/agreements/signed

The formatted response MUST contain the level of attestation used (server trusted or cryptographic with a DID), and the contents of the machine-readable agreement. Retrieving signed agreements MUST be completed via a GET, and the response MUST be in the format of:

```
{
  "signed_agreements": [
    {
      "agreement": {
        "version": <machine-readable agreement version>,
        "references": [
```

```
            "<link to hashed agreements>"
          ],
          "agreementId": "<UUID representing this agreement>",
          "created": <epoch time>,
          "ids": [
            "<optional DID id>"
          ],
          "permitted": [
            "<purpose>"
          ],
          "prohibited": [
            "<prohibition>"
          ],
          "validRoles": [
            "<role>"
          ]
        },
        "signature_type": "cryptographic",
        "signatures": [
          {
            "version": 1,
            "id": "did:myterms:fedid.myterms.info:iuK...",
            "signedOn": 1761841201,
            "type": "JWS/JCS",
            "jws": "eyJ...ICg"
          }
        ]
      },
      {
        "agreement": {
          "version": <machine-readable agreement version>,
          "references": [
            "<link to hashed agreements>"
          ],
          "agreementId": "<UUID representing this agreement>",
          "created": <epoch time>,
          "ids": [
            "<optional DID id>"
          ],
          "permitted": [
            "<purpose>"
          ],
          "prohibited": [
            "<prohibition>"
          ],
          "validRoles": [
            "<role>"
          ]
```

```
        },
        "signature_type": "server-trusted"
      }
    ]
  }
```

Server trusted signed agreements will not contain a signature key, as these agreements were signed using standard server trusted check boxes.

## 5. Discovery and standard response

### 5.1. Capability discovery

Per [RFC8615], all endpoints SHOULD be discoverable via a "/.well-known/" entry. The URL format for this MUST be:

https://<domain>.<tld>/.well-known/myterms-configuration

The response MUST be JSON in the format of:

```
{
  "methods": [
    "continuous",
    "on-demand"
  ],
  "get_agreement_endpoint": "https://.../agreements",
  "get_agreement_prefs_endpoint": "https://.../prefs",
  "get_agreement_signed_endpoint": "https://.../signed",
  "post_agreement_prefs_endpoint": "https://.../prefs",
}
```

Endpoint and method definitions are OPTIONAL based on what the system supports.

### 5.2. Standard responses to API requests

The following standard responses SHOULD be used on all API requests.

| HTTP Response Code | Description |
|---|---|
| 200 | Success when retrieving or setting any data |
| 401 | The user is not authenticated (invalid or missing token) |
| 403 | The user is authenticated but not allowed to perform the action |
| 404 | The requested user or agreements record doesn’t exist |
| 500 | Unknown general internal server error |
| 502 | Bad gateway |
| 503 | Service unavailable |
| 504 | Gateway timeout |

Table 4

## 6. IANA Considerations

This memo includes no request to IANA.

## 7. Security Considerations

This document should not affect the security of the Internet.

## 8. References

### 8.1. Normative References

[DID] W3C, "Decentralized Identifiers (DIDs) v1.0 - Core architecture, data model, and representations", 2022, <https://www.w3.org/TR/did-1.0/>.

[DPV] W3C, "Data Privacy Vocabulary", 2026, <https://w3c.github.io/dpv/2.3/dpv/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/rfc/rfc7515>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/rfc/rfc8615>.

[RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, DOI 10.17487/RFC8785, June 2020, <https://www.rfc-editor.org/rfc/rfc8785>.

## 8.2. Informative References

[IEEE7012] IEEE, "IEEE 7012 - Machine Readable Personal Privacy Terms", 2025, <https://ieeexplore.ieee.org/servlet/opac?punumber=11170391>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <https://www.rfc-editor.org/rfc/rfc7258>.

## Author’s Address

Benjamin Curtis
MyTerms
Email: ietf@nowsci.com